UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/523,877	03/13/2000	Peter Warnes	ARC.005A	6131

27299 7590 10/26/2004

GAZDZINSKI & ASSOCIATES
11440 WEST BERNARDO COURT, SUITE 375
SAN DIEGO, CA 92127

EXAMINER	
HUISMAN, DAVID J	
ART UNIT	PAPER NUMBER
2183	

DATE MAILED: 10/26/2004

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	09/523,877	WARNES ET AL.	
	Examiner	Art Unit	
	David J. Huisman	2183	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *27 July 2004*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *37-40,42,43 and 53-63* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *37-40,42,43 and 53-63* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *13 March 2000* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

1. Claims 37-40, 42-43, and 53-63 have been examined.

Papers Submitted

2. It is hereby acknowledged that the following papers have been received and placed of record in the file: Amendment as received on 7/27/2004.

Claim Rejections - 35 USC § 112

3. The following is a quotation of the second paragraph of 35 U.S.C. 112:

> The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

4. Claims 37-40, 42-43, 53-58, 60-61, and 63 are rejected under 35 U.S.C. 112, second paragraph, for having lack of antecedent basis issues within the claims.

5. Claim 37 recites the limitation "said pipeline". There is insufficient antecedent basis for this limitation in the claim because even though a "...pipelined digital RISC processor" is claimed, there is no explicit mention of a single pipeline. It is possible that a pipelined processor has multiple pipelines and this would interfere with "said pipeline".

6. Claim 38 recites the limitation "said pipeline". There is insufficient antecedent basis for this limitation in the claim because even though a "...pipelined digital RISC processor" is claimed, there is no explicit mention of a single pipeline. It is possible that a pipelined processor has multiple pipelines and this would interfere with "said pipeline".

7. Claim 39 recites the limitation "the particular combinations" and the limitation "the logical functions". There is insufficient antecedent basis for these limitations in the claim.

Claim 39 also recites the limitation "said pipeline". There is insufficient antecedent basis for these limitations in the claim because even though a "...pipelined digital RISC processor" is claimed, there is no explicit mention of a single pipeline. It is possible that a pipelined processor has multiple pipelines and this would interfere with "said pipeline".

8. Claim 40 recites the limitation "said instruction set" in line 7. There is insufficient antecedent basis for this limitation in the claim as it is not clear whether applicant is referring to the base instruction set or the extension instruction set. Claim 40 also recites the limitation "said pipeline". There is insufficient antecedent basis for these limitations in the claim because even though a "...pipelined digital RISC processor" is claimed, there is no explicit mention of a single pipeline. It is possible that a pipelined processor has multiple pipelines and this would interfere with "said pipeline".

9. Claim 42 recites the limitation "said at least four modes". There is insufficient antecedent basis for this limitation in the claim. Please replace "said at least four modes" with --said four modes--. Claim 42 also recites the limitation "said pipeline". There is insufficient antecedent basis for these limitations in the claim because even though a "...pipelined digital RISC processor" is claimed, there is no explicit mention of a single pipeline. It is possible that a pipelined processor has multiple pipelines and this would interfere with "said pipeline".

10. Claim 43 recites the limitation "said at least four modes". There is insufficient antecedent basis for this limitation in the claim. Please replace "said at least four modes" with --said four modes--.

11. Claim 53 recites the limitation "said bits of said branch instruction". There is insufficient antecedent basis for this limitation in the claim, as claim 37 does not establish that the bits are

encoded in the branch instruction. More specifically, claim 37 states that the bits determine the modes but does not state that the bits are encoded within the branch instruction.

12. Claim 54 recites the limitation "said bits of said branch instruction". There is insufficient antecedent basis for this limitation in the claim, as claim 37 does not establish that the bits are encoded in the branch instruction. More specifically, claim 37 states that the bits determine the modes but does not state that the bits are encoded within the branch instruction. Claim 54 also recites the limitation "the particular combinations" and the limitation "the logical functions". There is insufficient antecedent basis for these limitations in the claim.

13. Claim 55 recites the limitation "said bits of said branch instruction". There is insufficient antecedent basis for this limitation in the claim, as claims 37 and 53 do not establish that the bits are encoded in the branch instruction. More specifically, claim 37 states that the bits determine the modes but does not state that the bits are encoded within the branch instruction. Claim 55 also recites the limitation "the particular combinations" and the limitation "the logical functions". There is insufficient antecedent basis for these limitations in the claim.

14. Claim 56 recites the limitation "said plurality of bits". There is insufficient antecedent basis for these limitations in the claim.

15. Claim 57 recites the limitation "said bits of said branch instruction". There is insufficient antecedent basis for this limitation in the claim, as claim 38 does not establish that the bits are encoded in the branch instruction. More specifically, claim 38 states that the bits determine the modes but does not state that the bits are encoded within the branch instruction. Claim 54 also recites the limitation "the particular combinations" and the limitation "the logical functions". There is insufficient antecedent basis for these limitations in the claim.

16. Claim 58 recites the limitation "said bits of said branch instruction". There is insufficient antecedent basis for this limitation in the claim, as claim 38 does not establish that the bits are encoded in the branch instruction. More specifically, claim 38 states that the bits determine the modes but does not state that the bits are encoded within the branch instruction. Claim 54 also recites the limitation "the particular combinations" and the limitation "the logical functions". There is insufficient antecedent basis for these limitations in the claim.

17. Claim 60 recites the limitation "said pipeline". There is insufficient antecedent basis for this limitation in the claim because even though a "...pipelined digital RISC processor" is claimed, there is no explicit mention of a single pipeline. It is possible that a pipelined processor has multiple pipelines and this would interfere with "said pipeline".

18. Claim 61 recites the limitation "said pipeline". There is insufficient antecedent basis for this limitation in the claim because even though a "...pipelined digital RISC processor" is claimed, there is no explicit mention of a single pipeline. It is possible that a pipelined processor has multiple pipelines and this would interfere with "said pipeline".

19. Claim 63 recites the limitation "the method". There is insufficient antecedent basis for this limitation in the claim.

Claim Rejections - 35 USC § 103

20. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

21. Claims 37-40, 42-43, and 53-58 are rejected under 35 U.S.C. 103(a) as being unpatentable over Lee, as applied above, in view of Wirthlin et al., The Nano Processor: a Low Resource Reconfigurable Processor, 1994 (herein referred to as Wirthlin). In addition, Heuring and Jordan, "Computer Systems Design and Architecture," 1997 (herein referred to as Heuring) is cited as extrinsic evidence for providing a definition of RISC.

22. Referring to claim 37, Lee has taught a pipelined digital processor comprising:

a) a branch instruction having a plurality of user-configurable modes determined by a plurality of bits controlling the execution of at least one instruction in a delay slot following the branch instruction within the pipeline, each of said modes being constrained to only one of a plurality of unique combinations of said plurality of bits. See Fig.3 and note the configurable modes, which are specified by the nullify and sign/displacement bits (Fig.2, field 507). By setting or resetting these bits, the user will configure the system to either never nullify a delay slot instruction or sometimes nullify a delay slot instruction. One basic breakdown of the modes is as follows (modes are in the form (nullify/sign)):

00 - conditional branch forward with delay slot execution (FDS)
01 - conditional branch backwards with delay slot execution (BDS)
10 - don't branch and execute delay slot (DBE) or branch and nullify delay slot (BND)
11 - don't branch and nullify delay slot (DBN) or branch and execute delay slot (BED)

It should be realized that each combination of bits corresponds to a different mode. Therefore, each mode is constrained to only one unique combination of said plurality of bits.

b) Lee has not explicitly taught an extended and user-configured RISC processor having an instruction set that comprises a base instruction set and at least one extension instruction.

However, Wirthlin has taught a processor in which a base instruction set is used and an extended, custom instruction set (extension instruction) is used. See page 25, sections 3.1.1 and 3.1.2. Furthermore, it should be realized that Wirthlin has taught a RISC processor because as defined by Heuring on page 93 (see attached), RISC (reduced instruction set computer) machines focus on reducing the number of instructions in the machine. That is, a reduced instruction set exists. Clearly, this is the case in Wirthlin because only 6 standard instructions initially exist, and these instructions are hardly complex as they include load, store, add, subtract, and jump, which as is known in the art, are very common, standard instructions. See section 3.3. In addition, from the same section, Wirthlin has disclosed that the instructions are of a fixed length, which is another feature of a RISC machine (see page 93 of Heuring). Consequently, it can be seen that Wirthlin has taught an extended and user-configurable RISC processor. As disclosed by Wirthlin, the base instruction set comprises only the essential instructions while the extension instructions allow for the development of high-speed custom processors which would be able to perform the function desired by the user. As a result, in order to achieve custom processors, built specifically for a particular task, it would have been obvious to one of ordinary skill in the art at the time of the invention to implement a base instruction set and at least one extension instruction in a RISC processor. This will also maximize the efficiency and reduce the complexity of the system in that the processor will only have as many instructions as is required to perform the particular task. One would have been motivated to make such a combination because Lee could very well be a RISC processor, as it is not limited to a certain type of processor, and Lee has disclosed that instructions are single-cycle instructions (Fig.5 of Lee). And, this feature is a feature of a RISC machine (see page 93 of Heuring). Consequently, since Lee may be a RISC processor (less

complex according to Heuring), and Wirthlin has taught that extending a RISC processor is useful, such a combination would have been obvious.

23. Referring to claim 38, Lee has taught a pipelined digital processor comprising:

a) a branch instruction including two data bits defining four discrete modes controlling the execution of at least one instruction in a delay slot following the branch instruction within the pipeline. See the nullify and sign/displacement bits from Fig.3. These two bits define four delay sot modes as shown by the following breakdown (modes are in the form (nullify/sign)):

00 - conditional branch forward with delay slot execution (FDS)
01 - conditional branch backwards with delay slot execution (BDS)
10 - don't branch and execute delay slot (DBE) or branch and nullify delay slot (BND)
11 - don't branch and nullify delay slot (DBN) or branch and execute delay slot (BED)

It should be realized that each combination of bits corresponds to a mode with unique functionality with respect to the other three modes. Finally, see column 3, lines 58-61. By setting or clearing this nullify bit, a subsequent instruction's execution is controlled.

b) Lee has not explicitly taught an extended and user-configured RISC processor having an instruction set that comprises a base instruction set and at least one extension instruction. However, Wirthlin has taught a processor in which a base instruction set is used and an extended, custom instruction set (extension instruction) is used. See page 25, sections 3.1.1 and 3.1.2. Furthermore, it should be realized that Wirthlin has taught a RISC processor because as defined by Heuring on page 93 (see attached), RISC (reduced instruction set computer) machines focus on reducing the number of instructions in the machine. That is, a reduced instruction set exists. Clearly, this is the case in Wirthlin because only 6 standard instructions initially exist, and these instructions are hardly complex as they include load, store, add, subtract, and jump, which as is

known in the art, are very common, standard instructions. See section 3.3. In addition, from the same section, Wirthlin has disclosed that the instructions are of a fixed length, which is another feature of a RISC machine (see page 93 of Heuring). Consequently, it can be seen that Wirthlin has taught an extended and user-configurable RISC processor. As disclosed by Wirthlin, the base instruction set comprises only the essential instructions while the extension instructions allow for the development of high-speed custom processors which would be able to perform the function desired by the user. As a result, in order to achieve custom processors, built specifically for a particular task, it would have been obvious to one of ordinary skill in the art at the time of the invention to implement a base instruction set and at least one extension instruction in a RISC processor. This will also maximize the efficiency and reduce the complexity of the system in that the processor will only have as many instructions as is required to perform the particular task. One would have been motivated to make such a combination because Lee could very well be a RISC processor, as it is not limited to a certain type of processor, and Lee has disclosed that instructions are single-cycle instructions (Fig.5 of Lee). And, this feature is a feature of a RISC machine (see page 93 of Heuring). Consequently, since Lee may be a RISC processor (less complex according to Heuring), and Wirthlin has taught that extending a RISC processor is useful, such a combination would have been obvious.

c) said execution is controlled without regard to a branch direction metric. The examiner agrees that when the nullify bit is on, the branch direction is taken into consideration. However, when the nullify bit is off, execution is controlled without regard to the branch direction. Therefore, the claim is still anticipated.

24. Referring to claim 39, Lee has taught a pipelined digital processor comprising:

a) a branch instruction having at least one mode controlling the execution of at least one instruction in a delay slot following the branch instruction within the pipeline using a plurality of data bits, at least one of the particular combinations of data bits and the logical functions associated therewith being adapted for assignment by a user. As shown in Fig.3, there are a plurality of modes in which each branch can operate. The mode is dependent on the nullify bit and the displacement bit. Also, it should be realized that the user that is writing the program will be setting the nullify bit as well as the displacement bit by choosing whether the branch will be a forward or backwards branch. Therefore, they, along with the logical functions associated with the mode bits, are adapted for assignment by a user. Furthermore, this limitation can be interpreted as a user being able to assign a particular combination to a branch instruction, which indeed it taught by Lee, as each branch instruction will have a combination assigned to it.

b) Lee has not explicitly taught an extended and user-configured RISC processor having an instruction set that comprises a base instruction set and at least one extension instruction. However, Wirthlin has taught a processor in which a base instruction set is used and an extended, custom instruction set (extension instruction) is used. See page 25, sections 3.1.1 and 3.1.2. Furthermore, it should be realized that Wirthlin has taught a RISC processor because as defined by Heuring on page 93 (see attached), RISC (reduced instruction set computer) machines focus on reducing the number of instructions in the machine. That is, a reduced instruction set exists. Clearly, this is the case in Wirthlin because only 6 standard instructions initially exist, and these instructions are hardly complex as they include load, store, add, subtract, and jump, which as is known in the art, are very common, standard instructions. See section 3.3. In addition, from the same section, Wirthlin has disclosed that the instructions are of a fixed length, which is another

feature of a RISC machine (see page 93 of Heuring). Consequently, it can be seen that Wirthlin has taught an extended and user-configurable RISC processor. As disclosed by Wirthlin, the base instruction set comprises only the essential instructions while the extension instructions allow for the development of high-speed custom processors which would be able to perform the function desired by the user. As a result, in order to achieve custom processors, built specifically for a particular task, it would have been obvious to one of ordinary skill in the art at the time of the invention to implement a base instruction set and at least one extension instruction in a RISC processor. This will also maximize the efficiency and reduce the complexity of the system in that the processor will only have as many instructions as is required to perform the particular task. One would have been motivated to make such a combination because Lee could very well be a RISC processor, as it is not limited to a certain type of processor, and Lee has disclosed that instructions are single-cycle instructions (Fig.5 of Lee). And, this feature is a feature of a RISC machine (see page 93 of Heuring). Consequently, since Lee may be a RISC processor (less complex according to Heuring), and Wirthlin has taught that extending a RISC processor is useful, such a combination would have been obvious.

25. Referring to claim 40, Lee has taught a pipelined digital processor design comprising:

a) a branch instruction having four discrete modes controlling the execution of at least one instruction in a delay slot following the branch instruction within the pipeline, each of said modes provides unique functionality with respect to the other three modes. See the nullify and displacement bits from Fig.3. These two bits define four delay sot modes as shown by the following breakdown (modes are in the form (nullify/sign)):

00 - conditional branch forward with delay slot execution (FDS)

01 - conditional branch backwards with delay slot execution (BDS)
10 - don't branch and execute delay slot (DBE) or branch and nullify delay slot (BND)
11 - don't branch and nullify delay slot (DBN) or branch and execute delay slot (BED)

It should be realized that each combination of bits corresponds to a mode with unique functionality with respect to the other three modes. Finally, see column 3, lines 58-61. By setting or clearing this nullify bit, a subsequent instruction's execution is controlled.

b) Lee has not explicitly taught an extended and user-configured RISC processor having an instruction set that comprises a base instruction set and at least one extension instruction. However, Wirthlin has taught a processor in which a base instruction set is used and an extended, custom instruction set (extension instruction) is used. See page 25, sections 3.1.1 and 3.1.2. Furthermore, it should be realized that Wirthlin has taught a RISC processor because as defined by Heuring on page 93 (see attached), RISC (reduced instruction set computer) machines focus on reducing the number of instructions in the machine. That is, a reduced instruction set exists. Clearly, this is the case in Wirthlin because only 6 standard instructions initially exist, and these instructions are hardly complex as they include load, store, add, subtract, and jump, which as is known in the art, are very common, standard instructions. See section 3.3. In addition, from the same section, Wirthlin has disclosed that the instructions are of a fixed length, which is another feature of a RISC machine (see page 93 of Heuring). Consequently, it can be seen that Wirthlin has taught an extended and user-configurable RISC processor. As disclosed by Wirthlin, the base instruction set comprises only the essential instructions while the extension instructions allow for the development of high-speed custom processors which would be able to perform the function desired by the user. As a result, in order to achieve custom processors, built specifically for a particular task, it would have been obvious to one of ordinary skill in the art at the time of

the invention to implement a base instruction set and at least one extension instruction in a RISC processor. This will also maximize the efficiency and reduce the complexity of the system in that the processor will only have as many instructions as is required to perform the particular task. One would have been motivated to make such a combination because Lee could very well be a RISC processor, as it is not limited to a certain type of processor, and Lee has disclosed that instructions are single-cycle instructions (Fig.5 of Lee). And, this feature is a feature of a RISC machine (see page 93 of Heuring). Consequently, since Lee may be a RISC processor (less complex according to Heuring), and Wirthlin has taught that extending a RISC processor is useful, such a combination would have been obvious.

26. Referring to claim 42, Lee in view of Wirthlin has taught a digital processor as described in claim 40. Lee has further taught that first and second of said at least four modes implement one- and two-cycle stalls within said pipeline, respectively. See column 5, lines 32-37, and Fig.2. Note that if a jump occurs and the displacement is positive (as shown in Fig.2, component 112), the delay slot instruction would be fetched but not executed. Therefore, in order to kill the unwanted instruction, the pipeline would be stalled for at least a single cycle. Also, see Fig.5, column 5, lines 58-63, and column 7, lines 21-25. Note that pipeline includes four stages: an address generation stage (A), a fetch stage (F), an execution stage (E), and a write stage (W). It has been disclosed that the target address of the branch is not determined until the end of the execution stage. By this time, the delay slot instruction will have been fetched, and the predicted target address will be supplied to the program counter for fetching in the next cycle. See column 7, lines 6-10. If the delay slot instruction ends up being nullified (based on the displacement and

nullify bits) and the predicted target is incorrect, then both instructions will need to be cancelled, resulting in a 2-cycle stall.

27. Referring to claim 43, Lee in view of Wirthlin has taught a digital processor as described in claim 42. Lee has further taught that at least one of said at least four modes operates without respect to a branch displacement metric. See Fig.3 and note that when the nullify bit is off, the displacement value has no part in determining that the delay slot instruction is executed.

28. Referring to claim 53, Lee in view of Wirthlin has taught a processor as described in claim 37. Lee has further taught that:

a) said bits of said branch instruction comprise two data bits defining four discrete modes controlling said execution. See Fig.3 and note the configurable modes, which are specified by the nullify and sign/displacement bits (Fig.2, field 507). By setting or resetting these bits, the user will configure the system to either never nullify a delay slot instruction or sometimes nullify a delay slot instruction. One basic breakdown of the modes is as follows (modes are in the form (nullify/sign)):

00 - conditional branch forward with delay slot execution (FDS)
01 - conditional branch backwards with delay slot execution (BDS)
10 - don't branch and execute delay slot (DBE) or branch and nullify delay slot (BND)
11 - don't branch and nullify delay slot (DBN) or branch and execute delay slot (BED)

It should be realized that each combination of bits corresponds to a different mode.

b) said execution further being controlled without regard to a branch direction metric. When the nullify bit is off, execution is controlled without regard to the branch direction.

29. Referring to claim 54, Lee in view of Wirthlin has taught a processor as described in claim 37. Lee has further taught that at least one of the particular combinations of said bits of

said branch instruction and the logical functions associated therewith are adapted for assignment by a user. It should be noted that at the very least, the user will be specifying whether the branch is a forward or backwards branch, and therefore, the user is also specifying the displacement bit of the mode control bits. By doing this, the user is also specifying what functionality is to occur with such a displacement bit. Furthermore, this limitation can be interpreted as a user being able to assign a particular combination to a branch instruction, which indeed is taught by Lee, as each branch instruction will have a combination assigned to it.

30. Referring to claim 55, Lee in view of Wirthlin has taught a processor as described in claim 53. Lee has further taught that at least one of the particular combinations of said bits of said branch instruction and the logical functions associated therewith are adapted for assignment by a user. It should be noted that at the very least, the user will be specifying whether the branch is a forward or backwards branch, and therefore, the user is also specifying the displacement bit of the mode control bits. By doing this, the user is also specifying what functionality is to occur with such a displacement bit. Furthermore, this limitation can be interpreted as a user being able to assign a particular combination to a branch instruction, which indeed is taught by Lee, as each branch instruction will have a combination assigned to it.

31. Referring to claim 56, Lee in view of Wirthlin has taught a processor as described in claim 38. Lee has further taught that each of said four discrete modes is constrained to only one of a plurality of unique combinations of said plurality of bits. One basic breakdown of the modes is as follows (modes are in the form (nullify/sign)):

00 - conditional branch forward with delay slot execution (FDS)
01 - conditional branch backwards with delay slot execution (BDS)
10 - don't branch and execute delay slot (DBE) or branch and nullify delay slot (BND)

11 - don't branch and nullify delay slot (DBN) or branch and execute delay slot (BED)

It should be realized from this breakdown that each combination of bits corresponds to a different mode. Therefore, each mode is constrained to only one unique combination of said plurality of bits.

32. Referring to claim 57, Lee in view of Wirthlin has taught a processor as described in claim 38. Lee has further taught that at least one of the particular combinations of said bits of said branch instruction and the logical functions associated therewith are adapted for assignment by a user. It should be noted that at the very least, the user will be specifying whether the branch is a forward or backwards branch, and therefore, the user is also specifying the displacement bit of the mode control bits. By doing this, the user is also specifying what functionality is to occur with such a displacement bit. Furthermore, this limitation can be interpreted as a user being able to assign a particular combination to a branch instruction, which indeed is taught by Lee, as each branch instruction will have a combination assigned to it.

33. Referring to claim 58, Lee in view of Wirthlin has taught a processor as described in claim 56. Lee has further taught that at least one of the particular combinations of said bits of said branch instruction and the logical functions associated therewith are adapted for assignment by a user. It should be noted that at the very least, the user will be specifying whether the branch is a forward or backwards branch, and therefore, the user is also specifying the displacement bit of the mode control bits. By doing this, the user is also specifying what functionality is to occur with such a displacement bit. Furthermore, this limitation can be interpreted as a user being able to assign a particular combination to a branch instruction, which indeed is taught by Lee, as each branch instruction will have a combination assigned to it.

34. Claim 59 is rejected under 35 U.S.C. 103(a) as being unpatentable over Lee, as applied above, in view of Heuring, as applied above.

35. Referring to claim 59, Lee has taught a processor having at least one pipeline comprising at least instruction fetch, decode, and execute stages (see Fig.5 and column 6, lines 56-63, and note the explicit disclosure of fetch and execute stages. Although a decode stage is not explicitly mentioned, decoding of instructions is a step that must be performed. Decoding is disclosed in column 7, lines 16-20, and column 1, lines 23-26. Decoding will occur at some point in the pipeline and that will be the decode stage) and an associated data storage device, wherein the execution of instructions within said at least one pipeline is controlled by the method comprising: a) storing an instruction set within said data storage device, said instruction set comprising a plurality of instruction words, each of said instruction words comprising a plurality of data bits, at least one of said instruction words comprising a user-configurable branch instruction having a plurality of unique functional modes exclusively associated with respective ones of unique combinations of a plurality of mode control bits, said branch instruction directing branching to a first address within said data storage device. Lee has taught an instruction set with a plurality of instructions that would inherently comprise a plurality of bits and would inherently be stored in a data storage device in order to processor-accessible and executable. Furthermore, it is inherent that a branch will cause a jump to a specified address within the data storage device. Finally, it should be noted that these branch instructions are user-configurable instructions having unique functional modes associated with unique mode control bit combinations. For instance, one basic breakdown of the modes is as follows (modes are in the form (nullify/sign)):

00 - conditional branch forward with delay slot execution (FDS)
01 - conditional branch backwards with delay slot execution (BDS)
10 - don't branch and execute delay slot (DBE) or branch and nullify delay slot (BND)
11 - don't branch and nullify delay slot (DBN) or branch and execute delay slot (BED)

It should be realized that each combination of bits corresponds to a unique delay slot mode; that is, different functionality is achieved with each combination of bits.

b) assigning one of a plurality of values to each of said mode control bits of said at least one branch instruction. See column 3, lines 46-51. Lee discloses that each instruction contains 32 data bits of information. This information includes source registers, displacements, an opcode, condition fields, and a nullify bit and a displacement sign bit (mode control bits). Each of these bits is assigned a value (0 or 1). For instance, if the user decides to use a branch instruction, then he/she must set the nullify bit to the desired state and the displacement bit will be determined based on whether the user specifies the branch to be a forward or backwards branch.

c) decoding said at least one branch instruction including said assigned values. Since, the values of the nullify bit and displacement sign bit are part of each branch instruction, it follows that the value will be decoded as the branch instruction is decoded. See Fig.1 and column 3, lines 46-51.

d) determining whether to execute an instruction within said pipeline in a stage preceding that of said at least one branch instruction based at least in part on said assigned values. The delay slot instruction would be in a pipeline stage preceding that of the branch instruction. And, Lee discloses a system in which the execution of the delay slot instruction is determined based on the value of the nullify bit and/or displacement bit.

e) branching to said first address based on said at least one branching instruction. Recall from claim 16, that it is the inherent nature of a branch instruction (when taken) to jump to a specified address.

f) performing, based at least in part on said act of decoding said assigned values, at least one other function dictated by the unique functional mode associated with said assigned values. Looking at Fig.2, component 113, and Fig.3, it should be realized that depending on the mode of the branch instruction, the delay slot instruction will either be executed or not executed. If it is determined that the delay slot instruction will be executed, then it will eventually be performed following the branch. If the delay slot instruction is an "ADD" instruction, for instance, an addition function will be performed.

g) Lee has further taught that the processor is user-configured and extended. More specifically, it is user-configured in that the user may configure whether to nullify or not nullify delay slot instructions and it is extended because the processor provides the capability of allowing for the nullification of the delay slot instruction, i.e., the nullification is interpreted as an extended feature. Lee has not explicitly taught that the processor is a RISC processor. However, it should be noted that Lee does not say that the processor is not a RISC processor. Official Notice is taken that RISC processors are well known and expected in the art. In addition, Heuring has shown advantages of a RISC design on pages 93-94. Some of these advantages include a reduced instruction set which is also less complex, fixed instruction length, completion of execution every clock cycle, simplified addressing modes, prefetching, and speculative execution. As a result, in order to realize some of these advantages, it would have been obvious

to one of ordinary skill in the art at the time of the invention to modify Lee to be a RISC processor.

36. Claims 60-63 are rejected under 35 U.S.C. 103(a) as being unpatentable over Lee in view of Heuring, as applied above, and further in view of Kawasaki et al., U.S. Patent No. 5,530,965 (herein referred to as Kawasaki).

37. Referring to claim 60, Lee has taught a method of controlling the execution of instructions within a pipelined processor, comprising:

a) providing an instruction set comprising a plurality of instruction words. Lee discloses in column 3, lines 46-47, that each instruction within the instruction set contains a 6-bit opcode, which means a total of 64 instructions could exist within the system. Lee also discloses in column 6, lines 36-39, that the system contains a floating-point unit, which means floating-point instructions would exist.

b) each of said instruction words comprising a plurality of data bits. See column 3, lines 40-42. Each instruction is 32 bits.

c) at least one of said words comprising a jump instruction having at least one user-configurable mode and at least one user-definable mode associated therewith, said user-configurable and user-definable modes each being specified by the same ones of said plurality of data bits, said at least one user-definable mode not being predetermined in terms of function. See Fig.2, component 102, and note the use of a branch (jump) instruction. Also, it should be realized that that the user can configure a jump instruction such that its delay slot instruction is always executed by turning the nullify bit off (see Fig.3 and Fig.1, field 507). This is equivalent to at least one user-

configurable mode. In addition, the user can define the delay slot of an instruction to either execute or not execute by turning on the nullify bit (see Fig.3 and Fig.1, field 507). With the nullify bit on, the functionality is not predetermined because whether or not the delay slot instruction executes is partially dependent on whether the associated branch is taken or not taken, and a branch outcome is determined as the program is running. So the functionality will be determined while the program is running. This is equivalent to at least one user-definable mode. It can be seen that this configurability and definability are specified by the same bits (i.e., the bits specifically used by branch instructions - Fig.2, fields 507 and 508, but specifically, the nullify bit).

d) assigning one of a plurality of values to said ones of said data bits of said at least one jump instruction. See column 3, lines 46-51. Lee discloses that each branch instruction contains a nullify bit and a displacement sign bit. These bits can be assigned a value (0 or 1) as shown in Fig.3.

e) controlling the execution of at least one subsequent instruction within said pipeline based on said one assigned value of said ones of data bits when said at least one jump instruction is decoded. See column 3, lines 58-61. By setting or clearing this nullify bit, a subsequent instruction's execution is controlled.

f) wherein said method further comprises generating a long immediate constant using a single word instruction by:

f1) Lee has further taught of providing an instruction word having an opcode and at least one short immediate value associated therewith, said at least one short immediate value comprising a plurality of bits. Note from column 3, lines 46-51, that the instruction

format includes an 11 bit displacement field, which holds an immediate constant for branching purposes.

f2) Lee has not explicitly taught selecting a portion of said plurality of bits of said at least one short immediate value, shifting all of said bits of said at least one short immediate value using said opcode and only said portion of bits to produce a shifted immediate value, and storing said shifted immediate value in a register. However, Kawasaki has taught such a concept for branching purposes as well as for other instructions, such as move instructions. See column 51, lines 50-55. Kawasaki has disclosed a move instruction of the format: **mov #imm, Rn**, where the short immediate value specified by #imm is sign-extended to form a long immediate value which is then stored in the register specified by Rn. By sign-extending an immediate value, a portion (the sign bit) of the value is copied into the most significant bit positions of the long immediate value. For instance, if the #imm field specified a short 4-bit immediate value 1010, which is to be transformed into an 8-bit long immediate value, then 1010 would be sign-extended to 11111010, where the sign bit of the 4-bit value is copied into the 4 most significant bit positions of the long value. It also follows that the 4-bit value has been shifted. Note that initially, 1010 contained a 1 in the most significant bit position, a 0 in the second most significant bit position, a 1 in the third most significant bit position, and a 0 in the fourth most significant bit position. After sign-extending the 4-bit value, the same numbers become the fifth, sixth, seventh, and eighth most significant bits, respectively. Hence, they have been shifted. Furthermore, in column 42, lines 16-27, Kawasaki has disclosed that this type of move instruction is used to help branch to addresses out of the short

immediate value's range. More specifically, if a branch needs to branch further than what the short displacement allows, then the destination address is moved to the register specified by the "mov" instruction and a "jmp" instruction (shown in column 48, lines 28-30) is used with to jump to the address stored in the register. A person of ordinary skill in the art would have recognized that this concept could be applicable in a system that is concerned with branching, such as Lee's. Such a concept would allow a branch instruction to branch to an address outside of the range of just a short immediate displacement value. This in turn would give a programmer more freedom in that they would not have to worry about program length or certain parts of a program being out of reach of a branch. Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to take Kawasaki's concept of selecting a portion of said plurality of bits of said at least one short immediate value, shifting all of said bits of said at least one short immediate value using said opcode and only said portion of bits to produce a shifted immediate value, and storing said shifted immediate value in a register, and apply it to the system of Lee.

g) Lee has further taught that the processor is user-configured and extended. More specifically, it is user-configured in that the user may configure whether to nullify or not nullify delay slot instructions and it is extended because the processor provides the capability of allowing for the nullification of the delay slot instruction, i.e., the nullification is interpreted as an extended feature. Lee has not explicitly taught that the processor is a RISC processor. However, it should be noted that Lee does not say that the processor is not a RISC processor. Official Notice is taken that RISC processors are well known and expected in the art. In addition, Heuring has

shown advantages of a RISC design on pages 93-94. Some of these advantages include a reduced instruction set which is also less complex, fixed instruction length, completion of execution every clock cycle, simplified addressing modes, prefetching, and speculative execution. As a result, in order to realize some of these advantages, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Lee to be a RISC processor.

38. Referring to claim 61, Lee has taught a method of controlling the execution of instructions within a pipelined processor, comprising:

a) providing an instruction set comprising a plurality of instruction words. Lee discloses in column 3, lines 46-47, that each instruction within the instruction set contains a 6-bit opcode, which means a total of 64 instructions could exist within the system. Lee also discloses in column 6, lines 36-39, that the system contains a floating-point unit, which means floating-point instructions would exist.

b) each of said instruction words comprising a plurality of data bits. See column 3, lines 40-42. Each instruction is 32 bits.

c) at least one of said words comprising a jump instruction having at least one user-configurable mode and at least one user-definable mode associated therewith, said user-configurable and user-definable modes each being specified by the same ones of said plurality of data bits, said at least one user-definable mode not being predetermined in terms of function. See Fig.2, component 102, and note the use of a branch (jump) instruction. Also, it should be realized that that the user can configure a jump instruction such that its delay slot instruction is always executed by turning the nullify bit off (see Fig.3 and Fig.1, field 507). This is equivalent to at least one user-

configurable mode. In addition, the user can define the delay slot of an instruction to either execute or not execute by turning on the nullify bit (see Fig.3 and Fig.1, field 507). With the nullify bit on, the functionality is not predetermined because whether or not the delay slot instruction executes is partially dependent on whether the associated branch is taken or not taken, and a branch outcome is determined as the program is running. So the functionality will be determined while the program is running. This is equivalent to at least one user-definable mode. It can be seen that this configurability and definability are specified by the same bits (i.e., the bits specifically used by branch instructions - Fig.2, fields 507 and 508, but specifically, the nullify bit).

d) assigning one of a plurality of values to said ones of said data bits of said at least one jump instruction. See column 3, lines 46-51. Lee discloses that each branch instruction contains a nullify bit and a displacement sign bit. These bits can be assigned a value (0 or 1) as shown in Fig.3.

e) controlling the execution of at least one subsequent instruction within said pipeline based on said one assigned value of said ones of data bits when said at least one jump instruction is decoded. See column 3, lines 58-61. By setting or clearing this nullify bit, a subsequent instruction's execution is controlled.

f) wherein at least one of said plurality of instruction words comprises an op-code (column 3, lines 46-47), a plurality of fields (Fig.1), each of said fields comprising a plurality of bits (see column 3, lines 46-51), said at least one instruction word being encoded according to the method comprising:

f1) associating a first of said fields with a first data source. See column 3, lines 47-48 (component 503).

f2) associating a second of said fields with a second data source. See column 3, lines 48-49 (component 504).

f3) performing a logical operation using said first and second data sources as operands, said logical operation being specified by said op-code. See column 3, lines 51-55. In this case, the opcode specifies a compare and branch instruction where the comparison is performed between the contents of the two specified registers.

g) Lee has not taught that said method further comprises generating a long immediate constant using a single word instruction by selecting a portion of said plurality of bits of said at least one short immediate value, shifting all of said bits of said at least one short immediate value using said opcode and only said portion of bits to produce a shifted immediate value, and storing said shifted immediate value in a register. However, Kawasaki has taught such a concept for branching purposes as well as for other instructions, such as move instructions. See column 51, lines 50-55. Kawasaki has disclosed a move instruction of the format: **mov #imm, Rn**, where the short immediate value specified by #imm is sign-extended to form a long immediate value which is then stored in the register specified by Rn. By sign-extending an immediate value, a portion (the sign bit) of the value is copied into the most significant bit positions of the long immediate value. For instance, if the #imm field specified a short 4-bit immediate value 1010, which is to be transformed into an 8-bit long immediate value, then 1010 would be sign-extended to 11111010, where the sign bit of the 4-bit value is copied into the 4 most significant bit positions of the long value. It also follows that the 4-bit value has been shifted. Note that

initially, 1010 contained a 1 in the most significant bit position, a 0 in the second most significant bit position, a 1 in the third most significant bit position, and a 0 in the fourth most significant bit position. After sign-extending the 4-bit value, the same numbers become the fifth, sixth, seventh, and eighth most significant bits, respectively. Hence, they have been shifted. Furthermore, in column 42, lines 16-27, Kawasaki has disclosed that this type of move instruction is used to help branch to addresses out of the short immediate value's range. More specifically, if a branch needs to branch further than what the short displacement allows, then the destination address is moved to the register specified by the "mov" instruction and a "jmp" instruction (shown in column 48, lines 28-30) is used with to jump to the address stored in the register. A person of ordinary skill in the art would have recognized that this concept could be applicable in a system that is concerned with branching, such as Lee's. Such a concept would allow a branch instruction to branch to an address outside of the range of just a short immediate displacement value. This in turn would give a programmer more freedom in that they would not have to worry about program length or certain parts of a program being out of reach of a branch. Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to take Kawasaki's concept of selecting a portion of said plurality of bits of said at least one short immediate value, shifting all of said bits of said at least one short immediate value using said opcode and only said portion of bits to produce a shifted immediate value, and storing said shifted immediate value in a register, and apply it to the system of Lee.

h) Lee has further taught that the processor is user-configured and extended. More specifically, it is user-configured in that the user may configure whether to nullify or not nullify delay slot instructions and it is extended because the processor provides the capability of allowing for the

nullification of the delay slot instruction, i.e., the nullification is interpreted as an extended feature. Lee has not explicitly taught that the processor is a RISC processor. However, it should be noted that Lee does not say that the processor is not a RISC processor. Official Notice is taken that RISC processors are well known and expected in the art. In addition, Heuring has shown advantages of a RISC design on pages 93-94. Some of these advantages include a reduced instruction set which is also less complex, fixed instruction length, completion of execution every clock cycle, simplified addressing modes, prefetching, and speculative execution. As a result, in order to realize some of these advantages, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Lee to be a RISC processor.

39. Referring to claim 62, Lee has taught a processor comprising:

a) a processor core having a multistage instruction pipeline, said core being adapted to decode and execute an instruction set comprising a plurality of instruction words. Fig.5 shows a 4-stage pipeline that is further described in column 6, line 56, to column 7, line 39. Furthermore, it is inherent that the processor will decode and execute multiple instructions (as established in the rejection of claim 1) from an instruction set.

b) a data interface between said processor core and an information storage device. It is inherent that in order for a processor to execute instructions, they must be stored in the processor's memory. Therefore, the instructions would be stored in an information storage device that is directly accessible by the processor.

c) an instruction set comprising a plurality of instruction words, at least one of said instruction words being a user-configurable jump instruction containing data defining a plurality of jump

delay slot modes and at least one user-defined mode, said jump delay slot modes and at least one user-defined mode each being specified by the same portions of said data, said at least one user-defined mode not being predetermined in terms of function, said plurality of modes controlling the execution of instructions within said instruction pipeline of said processor core in response to said at least one jump instruction word within said instruction set. See Fig.2, component 102, and note the use of a branch (jump) instruction. Furthermore, Fig.3 shows that the branches are user-configurable in that a number of different modes which are achieved in part by the user. For example, the user can configure a jump instruction such that its delay slot instruction is always executed by turning the nullify bit off (see Fig.3 and Fig.1, field 507). In addition, the user can define the delay slot of an instruction to either execute or not execute by turning on the nullify bit (see Fig.3 and Fig.1, field 507). With the nullify bit on, the functionality is not predetermined because whether or not the delay slot instruction executes or not is partially dependent on whether the associated branch is taken or not taken, and a branch outcome is determined as the program is running. So the functionality will be determined while the program is running. This is equivalent to at least one user-definable mode. It can be seen that this configurability and definability are specified by the same bits (i.e., the bits specifically used by branch instructions - Fig.2, fields 507 and 508, but specifically, the nullify bit).

d) wherein said processor is further adapted to generate a long immediate constant using a single word instruction by:

f1) Lee has further taught of providing an instruction word having an opcode and at least one short immediate value associated therewith, said at least one short immediate value comprising a plurality of bits. Note from column 3, lines 46-51, that the instruction

format includes an 11 bit displacement field, which holds an immediate constant for branching purposes.

f2) Lee has not explicitly taught selecting a portion of said plurality of bits of said at least one short immediate value, shifting all of said bits of said at least one short immediate value using said opcode and only said portion of bits to produce a shifted immediate value, and storing said shifted immediate value in a register. However, Kawasaki has taught such a concept for branching purposes as well as for other instructions, such as move instructions. See column 51, lines 50-55. Kawasaki has disclosed a move instruction of the format: **mov #imm, Rn**, where the short immediate value specified by #imm is sign-extended to form a long immediate value which is then stored in the register specified by Rn. By sign-extending an immediate value, a portion (the sign bit) of the value is copied into the most significant bit positions of the long immediate value. For instance, if the #imm field specified a short 4-bit immediate value 1010, which is to be transformed into an 8-bit long immediate value, then 1010 would be sign-extended to 11111010, where the sign bit of the 4-bit value is copied into the 4 most significant bit positions of the long value. It also follows that the 4-bit value has been shifted. Note that initially, 1010 contained a 1 in the most significant bit position, a 0 in the second most significant bit position, a 1 in the third most significant bit position, and a 0 in the fourth most significant bit position. After sign-extending the 4-bit value, the same numbers become the fifth, sixth, seventh, and eighth most significant bits, respectively. Hence, they have been shifted. Furthermore, in column 42, lines 16-27, Kawasaki has disclosed that this type of move instruction is used to help branch to addresses out of the short

immediate value's range. More specifically, if a branch needs to branch further than what the short displacement allows, then the destination address is moved to the register specified by the "mov" instruction and a "jmp" instruction (shown in column 48, lines 28-30) is used with to jump to the address stored in the register. A person of ordinary skill in the art would have recognized that this concept could be applicable in a system that is concerned with branching, such as Lee's. Such a concept would allow a branch instruction to branch to an address outside of the range of just a short immediate displacement value. This in turn would give a programmer more freedom in that they would not have to worry about program length or certain parts of a program being out of reach of a branch. Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to take Kawasaki's concept of selecting a portion of said plurality of bits of said at least one short immediate value, shifting all of said bits of said at least one short immediate value using said opcode and only said portion of bits to produce a shifted immediate value, and storing said shifted immediate value in a register, and apply it to the system of Lee.

g) Lee has further taught that the processor is user-configured and extended. More specifically, it is user-configured in that the user may configure whether to nullify or not nullify delay slot instructions and it is extended because the processor provides the capability of allowing for the nullification of the delay slot instruction, i.e., the nullification is interpreted as an extended feature. Lee has not explicitly taught that the processor is a RISC processor. However, it should be noted that Lee does not say that the processor is not a RISC processor. Official Notice is taken that RISC processors are well known and expected in the art. In addition, Heuring has

shown advantages of a RISC design on pages 93-94. Some of these advantages include a reduced instruction set which is also less complex, fixed instruction length, completion of execution every clock cycle, simplified addressing modes, prefetching, and speculative execution. As a result, in order to realize some of these advantages, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Lee to be a RISC processor.

40. Referring to claim 63, Lee has taught a digital processor comprising:

a) a processor core having a multistage instruction pipeline, said core being adapted to decode and execute an instruction set comprising a plurality of instruction words. Fig.5 shows a 4-stage pipeline that is further described in column 6, line 56, to column 7, line 39. Furthermore, it is inherent that the processor will decode and execute multiple instructions (as established in the rejection of claim 1) from an instruction set.

b) a data interface between said processor core and an information storage device. It is inherent that in order for a processor to execute instructions, they must be stored in the processor's memory. Therefore, the instructions would be stored in an information storage device that is directly accessible by the processor.

c) an instruction set comprising a plurality of instruction words, at least one of said instruction words being a user-configurable jump instruction containing data defining a plurality of jump delay slot modes and at least one user-defined mode, said jump delay slot modes and at least one user-defined mode each being specified by the same portions of said data, said at least one user-defined mode not being predetermined in terms of function, said plurality of modes controlling the execution of instructions within said instruction pipeline of said processor core in response to

said at least one jump instruction word within said instruction set. See Fig.2, component 102, and note the use of a branch (jump) instruction. Furthermore, Fig.3 shows that the branches are user-configurable in that a number of different modes which are achieved in part by the user. For example, the user can configure a jump instruction such that its delay slot instruction is always executed by turning the nullify bit off (see Fig.3 and Fig.1, field 507). In addition, the user can define the delay slot of an instruction to either execute or not execute by turning on the nullify bit (see Fig.3 and Fig.1, field 507). With the nullify bit on, the functionality is not predetermined because whether or not the delay slot instruction executes or not is partially dependent on whether the associated branch is taken or not taken, and a branch outcome is determined as the program is running. So the functionality will be determined while the program is running. This is equivalent to at least one user-definable mode. It can be seen that this configurability and definability are specified by the same bits (i.e., the bits specifically used by branch instructions - Fig.2, fields 507 and 508, but specifically, the nullify bit).

d) wherein at least one of said plurality of instruction words comprises an op-code (column 3, lines 46-47), a plurality of fields (Fig.1), each of said fields comprising a plurality of bits (see column 3, lines 46-51), said at least one instruction word being encoded according to the method comprising:

d1) associating a first of said fields with a first data source. See column 3, lines 47-48 (component 503).

d2) associating a second of said fields with a second data source. See column 3, lines 48-49 (component 504).

d3) performing a logical operation using said first and second data sources as operands, said logical operation being specified by said op-code. See column 3, lines 51-55. In this case, the opcode specifies a compare and branch instruction where the comparison is performed between the contents of the two specified registers.

e) Lee has not taught that said at least one instruction word is used to generate a long immediate constant according to the method comprising selecting a portion of said plurality of bits of said at least one short immediate value, shifting all of said bits of said at least one short immediate value using said opcode and only said portion of bits to produce a shifted immediate value, and storing said shifted immediate value in a register. However, Kawasaki has taught such a concept for branching purposes as well as for other instructions, such as move instructions. See column 51, lines 50-55. Kawasaki has disclosed a move instruction of the format: **mov #imm, Rn**, where the short immediate value specified by #imm is sign-extended to form a long immediate value which is then stored in the register specified by Rn. By sign-extending an immediate value, a portion (the sign bit) of the value is copied into the most significant bit positions of the long immediate value. For instance, if the #imm field specified a short 4-bit immediate value 1010, which is to be transformed into an 8-bit long immediate value, then 1010 would be sign-extended to 11111010, where the sign bit of the 4-bit value is copied into the 4 most significant bit positions of the long value. It also follows that the 4-bit value has been shifted. Note that initially, 1010 contained a 1 in the most significant bit position, a 0 in the second most significant bit position, a 1 in the third most significant bit position, and a 0 in the fourth most significant bit position. After sign-extending the 4-bit value, the same numbers become the fifth, sixth, seventh, and eighth most significant bits, respectively. Hence, they have been shifted.

Furthermore, in column 42, lines 16-27, Kawasaki has disclosed that this type of move instruction is used to help branch to addresses out of the short immediate value's range. More specifically, if a branch needs to branch further than what the short displacement allows, then the destination address is moved to the register specified by the "mov" instruction and a "jmp" instruction (shown in column 48, lines 28-30) is used with to jump to the address stored in the register. A person of ordinary skill in the art would have recognized that this concept could be applicable in a system that is concerned with branching, such as Lee's. Such a concept would allow a branch instruction to branch to an address outside of the range of just a short immediate displacement value. This in turn would give a programmer more freedom in that they would not have to worry about program length or certain parts of a program being out of reach of a branch. Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention to take Kawasaki's concept of selecting a portion of said plurality of bits of said at least one short immediate value, shifting all of said bits of said at least one short immediate value using said opcode and only said portion of bits to produce a shifted immediate value, and storing said shifted immediate value in a register, and apply it to the system of Lee.

f) Lee has further taught that the processor is user-configured and extended. More specifically, it is user-configured in that the user may configure whether to nullify or not nullify delay slot instructions and it is extended because the processor provides the capability of allowing for the nullification of the delay slot instruction, i.e., the nullification is interpreted as an extended feature.

Response to Arguments

41. Applicant's arguments filed on July 27, 2004, have been fully considered but they are not persuasive.

42. Applicant argues the novelty/rejection of claims 37-40 on page 10 of the remarks, in substance that:

> "Hence, Wirthlin teaches a "general purpose" (e.g., CISC) core that has its core configuration fixed. In contrast, Applicant's claimed inventions of Claims 37-40 (and in fact, its commercial products embodying the claimed inventions) are both user-configurable (i.e., the core configuration can, and in many cases necessarily must, be altered or purposely chosen), and extensible (i.e., the core and its instruction set can be appended with extension instructions and associated hardware). None of the Examiner's cited art (especially Wirthlin) teaches such functionality. In fact, Wirthlin pointedly teaches away from Applicant's inventions of Claims 37-40, since Wirthlin requires that the core configuration be fixed as explicitly stated in the citation presented above."

43. These arguments are not found persuasive for the following reasons:

a) The examiner has been unable to find anywhere within Wirthlin that says that the core is a CISC core, as applicant alleges. Consequently, applicant cannot assume that it is not a RISC core, especially when the examiner can cite evidence, such as Heuring, which shows that RISCs have features which are displayed by Wirthlin. For instance, Wirthlin has taught fixed instruction lengths, which is a property of RISCs, and a reduced instruction set of only 6 standard instructions. This in itself makes Wirthlin a reduced set instruction computer as it has a reduced set of instructions. And, the core in Wirthlin is fixed. However, the processor as a whole (which includes the core) is user-configurable and extensible as custom instructions may be added into the system. This is enough to anticipate applicant's current claim language.

Conclusion

44. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure. Applicant is reminded that in amending in response to a rejection of claims, the patentable novelty must be clearly shown in view of the state of the art disclosed by the references cited and the objections made. Applicant must also show how the amendments avoid such references and objections. See 37 CFR § 1.111(c).

Portanova et al., U.S. Patent No. 4,992,934, has taught a reduced instruction computing apparatus and methods. More specifically, Portanova has taught a RISC architecture for executing RISC instructions (base set) and user-defined complex instructions (extended set).

Any inquiry concerning this communication or earlier communications from the examiner should be directed to David J. Huisman whose telephone number is (703) 305-7811. The examiner can normally be reached on Monday-Friday (8:00-4:30).

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Eddie Chan can be reached on (703) 305-9712. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

DJH
David J. Huisman
October 13, 2004

EDDIE CHAN
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100